Cykle wytwarzania oprogramowania

Celem procesu wytwarzania oprogramowania jest dostarczenie zamawiającemu spełniającego wymagania i sprawnego systemu oprogramowania. Z góry narzucone ograniczenia (np. czasowe, finansowe) oraz stopień złożoności problemu i technik jego produkcji wymuszają przedsięwzięcie odpowiednich czynności oraz właściwą ich organizację w planie realizacji projektu. Cechą każdego projektu mającego na celu stworzenie produktu jest ukierunkowanie wszystkich zadań objętych procesem wytwórczym na realizację wspólnego celu. W przypadku projektu informatycznego wspólnym celem jest wytworzenie systemu oprogramowania.

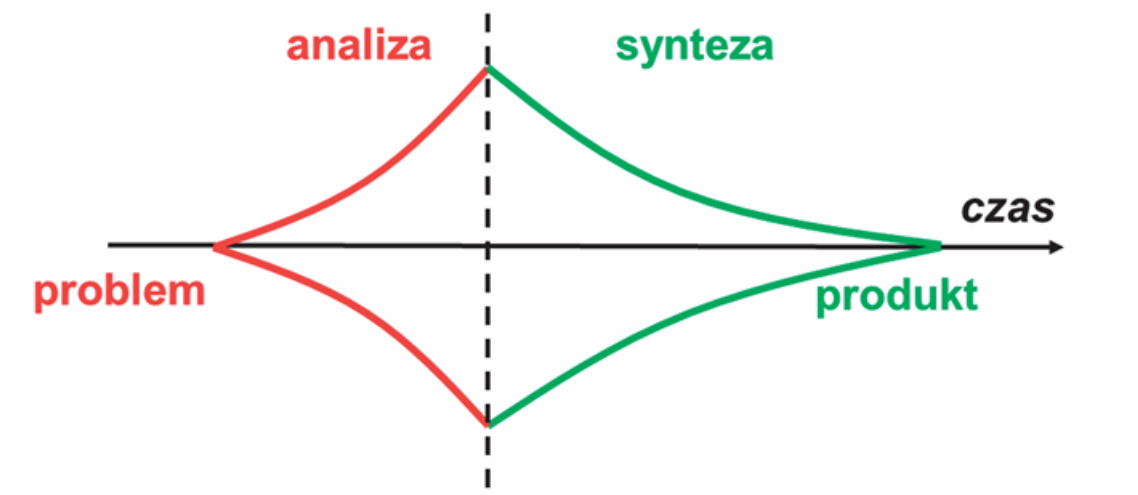
zakończenie projektu informatycznego sukcesem nie jest łatwym zadaniem. Różne stopnie skomplikowania problemów, często zmieniające się środowisko i wymagania na system oraz różne zasoby wymu szają indywidualne podejście do każdego projektu. Wytwarzanie oprogramowania można jednak przedstawić jako powtarzalny cykl rozwiązywania poszczególnych problemów, prac technicznych i łączenia rozwiązań. Na podstawie doświadczeń zebranych na przestrzeni lat opracowano wiele modeli procesów wytwórczych oprogramowania, które przedstawiają różne cykle wytwarzania oprogramowania.

<u>Dyscypliny cyklu wytwarzania oprogramowania</u>

W procesie wytwarzania oprogramowania powstaje przede wszystkim system oprogramowania. Oprócz tego w trakcie projektu konieczne jest wytworzenie innych produktów, takich jak instrukcja użytkownika, specyfikacja wymagań, dokumentacja projektowa itd. Takie produkty uzupełniające (zwane także pośrednimi) powstają w wyniku różnych działań, które możemy podzielić na grupy nazywane dyscyplinami. Dyscyplina inżynierii wymagań stanowi spójny zestaw czynności, które są powiązane z zasadniczym obszarem działania w ramach projektu konstrukcji oprogramowania. Podział na dyscypliny jest realizacją zasady dekompozycji problemu na mniejsze składniki, co ułatwia organizację pracy w projekcie.

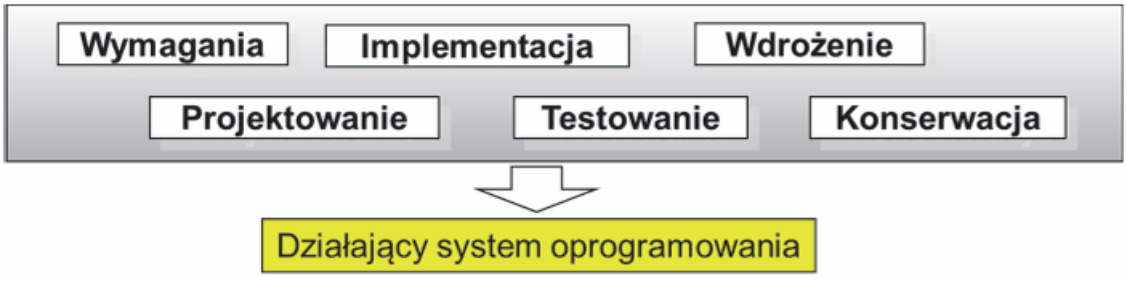
Jak dla każdej działalności inżynierskiej, czynności w inżynierii oprogramowania można podzielić na dwie grupy: czynności analizy i czynności syntezy. Taki podział jest naturalną dla człowieka metodą radzenia sobie ze skomplikowanymi zadaniami twórczymi: analiza polega na dokładnym zidentyfikowaniu i zrozumieniu problemu, którego rozwiązanie osiąga się poprzez syntezę, czyli realizację i scalanie mniejszych części.

Podążając za naturalną dla człowieka ścieżką analizy i syntezy (rysunek), od problemu do jego rozwiązania, proces wytwarzania oprogramowania definiuje się jako ciąg czynności podzielonych na dyscypliny, które prowadzą od postawienia problemu do wytworzenia produktu głównego (systemu oprogramowania) i produktów pośrednich. Do czynności analitycznych należą: opisanie środowiska, specyfikowanie wymagań, natomiast do czynności syntetycznych należą projektowanie, implementacja i wdrożenie. Rysunek obrazuje wzajemną zależność czynności analitycznych i syntetycznych. Produkty dyscyplin analitycznych, które prowadzą od ogółu do szczegółu, są podstawą do rozpoczęcia prac syntetycznych, które poprzez realizację i scalanie mniejszych części prowadzą do wytworzenia spójnego produktu.



Ogólne określenie procesu wytwarzania oprogramowania jako sekwencji analizy i syn tezy wyznacza jedynie ścieżkę od problemu do produktu. Praktyczne zastosowanie takiego podejścia inżynierskiego prowadzi do różnych rozwiązań w zakresie zdefiniowania i uporządkowania konkretnych zadań. Duża różnorodność projektów informatycznych wymaga bowiem dostosowywania kształtu procesu do indywidualnych potrzeb. W następnej sekcji niniejszego rozdziału zostaną przedstawione główne typy cykli wytwórczych w inżynierii oprogramowania. Najpierw jednak przedstawimy uporządkowanie zadań w ramach różnych cykli życia. Metodą na takie uporządkowanie jest podział na dyscypliny inżynierii oprogramowania. Należy przy tym zwrócić uwagę na to, że podział na dyscypliny nie oznacza podziału na etapy, czyli nie decyduje o uporządkowaniu czasowym wykonywania zadań w poszczególnych dyscyplinach. Takie uporządkowanie jest głównym elementem definicji cyklu życia.

Niezależnie od wybranego typu cyklu życia każdy projekt wytwarzania oprogramowania obejmuje kilka dyscyplin. Każda z dyscyplin dostarcza przynajmniej je den produkt (pośredni). Produkty pośrednie są punktami wyjścia dla czynności wykonywanych z tej lub innych dyscyplin. Na rysunku przedstawiono główne dyscypliny procesu wytwarzania oprogramowania w kolejności zgodnej ze ścieżką „analiza – synteza”.



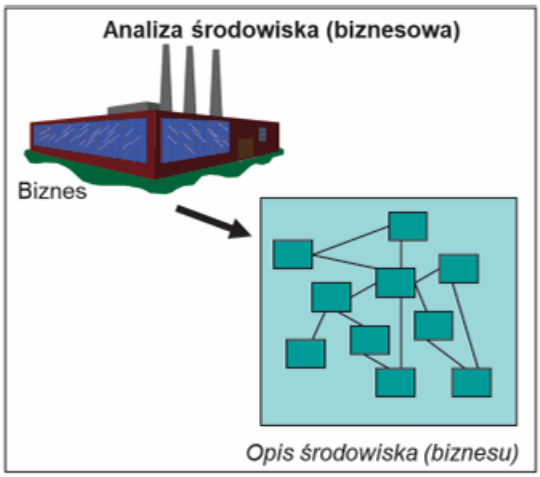
Dyscyplina wymagań dostarcza informacji o kształcie budowanego systemu z punktu widzenia klienta (np. użytkowników). Jest to określenie dość ogólne, albowiem na kształt systemu ma wpływ wiele czynników. Główne z nich to:

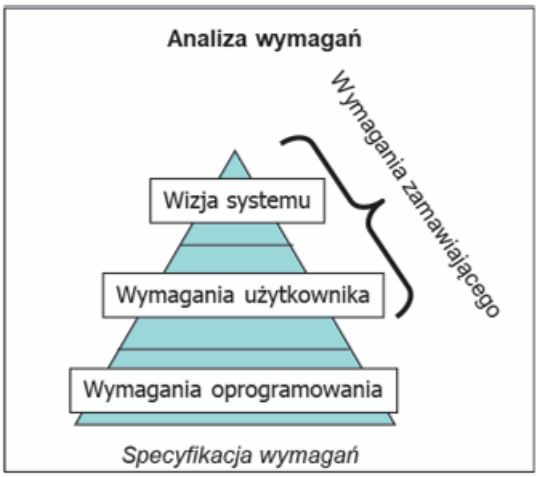
- środowisko, w którym system będzie funkcjonował,
- zadania, które system będzie realizował,
- sposób, w jaki system będzie funkcjonował.

Główne produkty dyscypliny wymagań to opis środowiska (np. środowiska biznesowego) i specyfikacja wymagań. Na podstawie specyfikacji wymagań dyscyplina projektowania wytwarza zbiór modeli projektowych (architektura, projekt bazy danych, projekty szczegółowe komponentów, projekty dynamiki działania komponentów). Dyscyplina implementacji realizuje założenia określone w modelach projektowych. Powstaje działający system, który jest gotowy do kontroli jakości w ramach dyscypliny testowania. Pomyślne przejście testów umożliwia wdrożenie systemu w środowisku produkcyjnym (dyscyplina wdrożenia), co obejmuje również m.in. przeszkolenie użytkowników i przygotowanie dokumentacji technicznej. Ostatecznym produktem jest działający system informatyczny, który powinien podlegać ciągłemu utrzymaniu w ramach dyscypliny konserwacji.

Dyscyplina wymagań

W ramach dyscypliny wymagań powinniśmy w pierwszej kolejności zidentyfikować problemy, które może rozwiązać system oprogramowania, a także podjąć decyzję o budowie takiego systemu. Podstawą określenia potrzeb klienta wynikających z zadanego problemu jest specyfikacja wymagań. Głównym celem specyfikacji wymagań jest określenie zakresu i kształtu przyszłego systemu, który będzie realizował potrzeby klienta. Rysunek przedstawia ogólny podział i strukturę dyscypliny wymagań. W ramach proponowanego przez nas podziału dyscyplina ta dzieli się na dwie zasadnicze dyscypliny składowe: analizę środowiska (np. biznesu) oraz analizę wymagań. Warto tutaj jednak podkreślić, że w niektórych metodykach te dwa składniki są traktowane jako osobne dyscypliny główne.





W ramach analizy środowiska opisujemy aktualny oraz docelowy stan środowiska (biznesu). Wykonywane jest to przy współpracy przedstawicieli klienta (osób zamawiających system lub ich reprezentantów). Na tej podstawie definiuje się główne potrzeby użytkowników systemu, które po uwzględnieniu ograniczeń środowiskowych tworzą wymagania zamawiającego. Poprawnie zdefiniowane wymagania zamawiającego zwykle są podstawą do zawarcia kontraktu między zamawiającym i wykonawcą na dostarczenie produktu spełniającego mieszczące się w tym dokumencie wymagania. Kolejną częścią specyfikacji wymagań jest opis wymagań oprogramowania, przedstawiający szczegóły działania systemu (m.in. szczegółowy opis interakcji systemu z użytkownikiem, wygląd okien).

Opis środowiska (biznesu) stanowi pierwszy zasadniczy produkt dyscypliny wymagań. Jest on opisem fragmentu świata, który jest obszarem działań danej organizacji wraz ze środowiskiem, w jakim ta organizacja (biznes) działa. Formułowanie takiego opisu polega na stworzeniu precyzyjnego i zrozumiałego modelu, który zawiera wszystkie elementy środowiska (biznesowego) oraz zasady oddziaływania ich na siebie. Taki model składa się zazwyczaj z setek lub nawet tysięcy różnych elementów oddziałujących na siebie w określony, złożony sposób. Podstawowe elementy organizacji biznesowej to współpracownicy, pracownicy, jednostki organizacyjne, produkty, surowce, podzespoły, dokumenty, systemy informatyczne. Jednym z tych elementów jest budowany system informatyczny. Modelowanie może przebiegać na różnym poziomie abstrakcji: możemy modelować całą firmę lub tylko jakiś jej fragment (np. dział czy placówkę). Model może opisywać tylko stan docelowy (po zbudowaniu systemu informatycznego) lub być uzupełniony o opis stanu aktualnego.

Opis środowiska zawiera zwykle słownik pojęć (biznesowych) reprezentujący strukturę środowiska (biznesowego) oraz składniki procesów (biznesowych). Procesy opisywane są z wykorzystaniem pojęć ze słownika pojęć. Procesy biznesowe są serią powiązanych ze sobą działań, które prowadzą do osiągnięcia celu biznesowego. Dokładna ich analiza dostarcza informacji na temat rzeczywistych potrzeb użytkowników, które są podstawą do zdefiniowania tzw. wymagań funkcjonalnych. Analiza środowiska, w którym realizowane są procesy biznesowe, dostarcza informacji na temat uwarunkowań środowiskowych, które są podstawą do sformułowania tzw. wymagań jakościowych (inaczej: pozafunkcjonalnych) na budowany system.

Kompletna **specyfikacja wymagań** wynika bezpośrednio z opisu biznesu i można ją porównać do piramidy Na samym szczycie znajduje się wizja systemu, która dostarcza informacji na temat ogólnych cech systemu w ścisłym powiązaniu z potrzebami biznesowymi klienta. Zakres systemu wyzna czają wymagania użytkownika w postaci uporządkowanego zbioru cech i funkcji systemu, które powinny być podstawą do zawarcia kontraktu. Wizja systemu i wymagania użytkownika tworzą wymagania zamawiającego, które są odzwierciedleniem rzeczywistych potrzeb zdefiniowanych przez biznes. Na najniższym poziomie piramidy są wymagania oprogramowania, które opisują szczegóły funkcjonalne komunikacji między użytkownikami i systemem, wszystkie wymieniane między nimi dane oraz planowany wygląd systemu. Specyfikacja wymagań, obok ogólnych cech systemu, zawiera opis jego dynamiki oraz struktury. Słownik pojęć (biznesowych) jest podstawą do stworzenia słownika dziedziny. Słownik dziedziny jest rozszerzany w trakcie opisywania cech systemu, wymagań funkcjonalnych i jakościowych. Słownik przyjmuje ostateczny kształt na koniec tworzenia specyfikacji wymagań. Zapewnia spójność całej specyfikacji wymagań i jest istotnym jej elementem. Na podstawie takiej spójnej i kompletnej specyfikacji wymagań można zbudować system informatyczny, będący naturalnym fragmentem rzeczywistości przedstawionej w opisie biznesu i realizujący rzeczywiste potrzeby zamawiającego.

Dyscyplina projektowania

W ramach dyscypliny projektowania dokonujemy syntezy problemu opisanego w ra mach dyscypliny wymagań poprzez opracowanie „planów projektowych" dla systemu. Jako produkt bazowy przyjmujemy zatem specyfikację wymagań i opis środowiska biznesu. Celem działań tej dyscypliny jest stworzenie modeli projektowych na różnych poziomach abstrakcji, na podstawie których zostanie zaimplementowany system. Projektowanie często zaczynamy od najbardziej ogólnych (abstrakcyjnych) modeli, a potem — w zależności od złożoności systemu — przechodzimy do bardziej szczegółowych poziomów (tzw. podejście top-down). Możliwe jest również podejście odwrotne, w którym zaczynamy od szczegółów, a dopiero potem syntetyzujemy widok ogólny systemu (tzw. podejście bottom-up).

Mając do dyspozycji opis środowiska biznesu i specyfikację wymagań, można zdefiniować, jakich elementów fizycznych (np. serwer bazy danych, serwer aplikacyjny, maszyny klienckie, urządzenia mobilne) będzie potrzebował budowany system i w jaki sposób te elementy będą się ze sobą komunikować. Jest to najbardziej ogólny, fizyczny model architektoniczny. Nie definiuje on logiki działania systemu, która definiowana jest na poziomie architektury logicznej. Na tym poziomie definiuje się moduły wykonawcze (komponenty), z jakich będzie się składał budowany system, oraz powiązania komunikacyjne (np. interfejsy) między tymi modułami. W często stosowanych technologiach opartych na usługach lub mikrousługach (ang. service, microservice) komponenty w ramach logicznego modelu architektonicznego reprezentują poszczególne usługi. Interfejsy, które są specyfikacją funkcji udostępnianych przez komponenty architektoniczne, pozwalają traktować moduły jako czarne skrzynki. Na tym poziomie abstrakcji nie skupiamy się na tym, „co jest w środku" komponentów. Skupiamy się na tym, jakie funkcje komponenty spełniają.

Istotnym elementem modelu architektonicznego i zależnych od niego modeli szczegó łowych są struktury danych, które służą do komunikacji poprzez interfejsy. Ich pod stawą może być słownik dziedziny, zdefiniowany w specyfikacji wymagań. Na pod stawie słownika możemy zdefiniować tzw. obiekty transferu danych (ang. data transfer object). Określają one „paczki danych", które mogą być przesyłane między poszczególnymi komponentami systemu.

Kolejnym poziomem projektowania jest stworzenie szczegółowych projektów modułów. Na tym poziomie definiujemy konkretne szczegóły „czarnych skrzynek". Dokładnie przedstawiamy strukturę oraz opracowujemy sekwencje wykonywanych operacji, których celem jest realizacja funkcji określonych w poszczególnych interfejsach komponentów. W zależności od złożoności systemu może być także wymagane zaprojektowanie algorytmów, które realizują złożone funkcjonalności wewnątrz systemu (np. algorytmy szyfrujące, obliczające składki ubezpieczenia czy wykonujące obliczenia na macierzach).

Produktem projektowania jest zbiór spójnych modeli dostarczających szczegółowych informacji o składnikach systemu, który ma spełniać wymagania opisane w specyfikacji wymagań i funkcjonować w opisanym środowisku. Modele projektowe bezpośrednio odzwierciedlają konstrukcje programistyczne i są bezpośrednio wykorzystywane podczas kodowania systemu, czyli jego implementacji.

Dyscyplina implementacji

W wyniku działań objętych dyscypliną implementacji zostaje wytworzony kod (program) realizujący wymagania zdefiniowane podczas analizy i spełniający założenia projektowe, określone w modelach projektowych. Do wytworzenia działającego systemu wykorzystuje się zbiór technologii, które są zgodne z wymaganiami technicznymi i wpisują się w środowisko działania budowanego systemu. Technologie te zostały również określone podczas projektowania komponentów systemu.

Proces programowania wymaga odpowiedniego środowiska i organizacji pracy. Aby stworzyć system, który będzie prawidłowo funkcjonował w środowisku produkcyjnym klienta, należy takie środowisko zbudować również w wersji testowej (implementacyjnej). Czynności z tym związane mogą obejmować na przykład instalację serwera i potrzebnych bibliotek w wersjach wykorzystywanych przez organizację, uruchomienie dedykowanych rozwiązań lub tzw. zaślepek potrzebnych do funkcjonowania systemu. Duży wpływ na kształt środowiska implementacyjnego mają wymagania jakościowe i ograniczenia środowiska biznesowego.

Implementacja systemu oznacza implementację poszczególnych modułów (komponentów) zaprojektowanych w ramach dyscypliny projektowania. Liczba modułów do zaimplementowania jest oczywiście zależna od stopnia skomplikowania systemu. Każdy moduł może być realizowany niezależnie od pozostałych (przez różnych programistów), ale powinien poprawnie z nimi współdziałać. W szczególności w architekturach komponentowych zadaniem programistów jest zaprogramowanie realizacji interfejsów dostarczanych przez komponenty, zgodnie z założeniami projektowymi.

Każdy zespół programistów realizujących pewną część systemu (zbiór modułów) jest odpowiedzialny za jej prawidłowe funkcjonowanie. Dlatego też nawet najmniejsze „kawałki" kodu powinny być na bieżąco testowane podczas całego procesu implementacji. Testowaniu podlegają poszczególne elementy kodu (klasy, funkcje), a także moduły (zestawy klas) oraz ich interfejsy (zestawy funkcji). Programiści tworzą tzw. testy jednostkowe. Są to zestawy procedur, których celem jest sprawdzenie

działania tworzonego kodu dla różnych sytuacji (dla różnych danych wejściowych). Zauważmy tutaj, że testy jednostkowe są traktowane jako element implementacji systemu, a nie jego walidacji

Prace w ramach dyscypliny implementacji wymagają zastosowania odpowiednich narzędzi, tworzących tzw. zintegrowane środowisko deweloperskie (ang. Integrated Development Environment — IDE). Podstawą jest oczywiście dobre narzędzie służące do kompilacji kodu, jego wykonywania w środowisku implementacyjnym (testowym) oraz do przeprowadzania testów jednostkowych. Inne narzędzia wspomagają pracę grupową, kontrolę wersji kodu oraz integrację całego systemu. Niezależnie jednak od stosowania dobrych narzędzi gwarancją wyprodukowania poprawnego i optymalnego kodu jest wykorzystanie dobrej znajomości technologii i dobrych praktyk programistycznych. Pożądaną cechą kodów źródłowych, oprócz oczywistej zgodności z modelami projektowymi i wymaganiami, powinny być: wspólna konwencja programistyczna, dobra dokumentacja i gwarancja poprawności działania potwierdzona dokumentacją testów jednostkowych.

Po implementacji sprawnie działających (przetestowanych) modułów i ich połączeniu całość powinna przejść testy integracyjne. Poprawność działania zostaje sprawdzona globalnie, dla całego systemu. Tak sprawdzony kod systemu jest głównym produktem dyscypliny implementacji. Pozostałe produkty to dokumentacja testów jednostkowych i integracyjnych oraz na przykład pliki wykonywalne i instalacyjne.

Dyscyplina walidacji (testowania)

Celem testowania w ramach osobnej dyscypliny walidacji jest sprawdzenie, czy zbudowany system oprogramowania spełnia wymagania określone w specyfikacji wymagań i działa poprawnie w środowisku biznesowym.

Testy, niezależnie od ich poziomu, najczęściej wykonuje się w specjalnie przygotowanym środowisku testowym. W takim środowisku system uruchamiany jest w sposób bardzo zbliżony do tego, jak będzie uruchamiany docelowo w tzw. środowisku produkcyjnym. W odróżnieniu od testów przeprowadzonych w czasie implementacji, głównymi odbiorcami testów w ramach walidacji są przedstawiciele zamawiającego. Zwykle są to przyszli użytkownicy systemu, wytypowani i specjalnie przygotowani do nadzorowania i wykonywania czynności związanych z testowa niem. Poza sprawdzeniem poprawności integracji, które dokonywane jest przez osoby odpowiedzialne za infrastrukturę informatyczną organizacji, sprawdzana jest również zgodność z wymaganiami zamawiającego oraz szczegółowymi wymaganiami oprogramowania. Ogół tego typu testów nazywamy testami akceptacyjnymi (ang. acceptance test), gdyż są one warunkiem akceptacji systemu przez klienta.

Przeprowadzenie testów akceptacyjnych systemu wymaga planu testów. Zazwyczaj plan testów jest dokumentem powstałym na podstawie specyfikacji wymagań. Poszczególne testy ułożone są w odpowiedniej kolejności, która pozwala na właściwe sprawdzenie prawidłowości działania poszczególnych funkcjonalności systemu. Testy takie przypominają instrukcje postępowania przygotowane dla użytkowników systemu. Rezultatem jest powstanie kompletnych scenariuszy testów. Efektem wykonania konkretnego scenariusza testów jest potwierdzenie popra ności działania lub opis błędu. W przypadku niepomyślnego wyniku testów może nastąpić powrót do działań w ramach innych dyscyplin (projektowanie, implementacja), a potem ponowienie testów po zaimplementowaniu poprawek. Wynikiem dyscypliny testowania jest dokumentacja testów

akceptacyjnych, która jest podstawą do zatwierdzenia systemu i przejścia do czynności objętych dyscypliną wdrożenia.

Dyscyplina wdrożenia

Wdrożenie systemu polega zasadniczo na przeniesieniu systemu do warunków środowiska produkcyjnego i przekazaniu go do właściwego użytkowania. Następuje to po pomyślnym przejściu testów akceptacyjnych i zatwierdzeniu uruchomienia aktualnej wersji budowanego systemu. Dyscyplina wdrożenia obejmuje czynności związane z instalacją systemu oraz umożliwieniem korzystania z systemu przez jego użytkowników. Instalacja systemu obejmuje zarówno przygotowanie środowiska produkcyjnego do wdrożenia nowego systemu, jak i samo fizyczne umieszczenie plików wykonywalnych systemu w środowisku docelowym. Głównym produktem dyscypliny wdrożenia jest zainstalowany system, działający w docelowym środowisku wykonawczym. Innymi produktami są na przykład dokumentacja dla użytkowników oraz dokumentacja techniczna. Strategie działań w ramach dyscypliny wdrożenia zależą od rodzaju budowanego systemu.

Można wyróżnić kilka najbardziej popularnych strategii wdrożenia nowego systemu do środowiska produkcyjnego:

- Wdrożenie bezpośrednie — nowy system zostaje wdrożony od razu w całości. Jest to strategia niosąca często spore ryzyko. Konieczne jest zapewnienie wysokiej jakości nowego systemu oprogramowania, gdyż usterki wykrywane w trakcie działania mogą spowodować duże problemy w organizacji klienta. W niektórych przypadkach, jak wdrożenie pierwszego systemu oprogramowania w danym obszarze czy też wymiana oprogramowania wbudowanego, taka strategia jest jedyną możliwą do zastosowania.
- Wdrożenie równoległe — obok nowego systemu przez pewien czas działa również stary. Zaletą tej strategii jest weryfikacja poprawności działania nowego systemu i możliwość wykorzystania starego w przypadku wykrycia błędów w nowym systemie lub jego awarii. Wadą jest natomiast duży koszt utrzymania i obsługi obu systemów.
- Wdrożenie pilotażowe — w pierwszej fazie fragment nowego systemu lub cały system zostaje uruchomiony w jednej z jednostek organizacyjnych. Dopiero po sprawdzeniu poprawności tego wdrożenia system zostaje wdrożony w pozostałych jednostkach. Takie podejście pozwala zminimalizować ryzyko związane z wdrożeniem bezpośrednim, a jednocześnie zredukować koszty utrzymania dwóch systemów.
- Wdrożenie stopniowe — strategia polegająca na wprowadzaniu w działalność organizacji kolejnych fragmentów nowego systemu, uniezależniając pozostałe obszary działalności od nowego systemu. Pozwala to zminimalizować ryzyka związane z zatrzymaniem działalności całej organizacji przez uruchomienie nowego systemu w całości. Wadą jest natomiast rozległe w czasie wdrożenie.

Warto zauważyć, że często wdraża się system według strategii hybrydowej, łączącej wybrane cechy powyższych strategii. Dzięki temu możliwe jest dostosowanie czynności wdrożeniowych do specyfiki danej organizacji i wdrażanego systemu oprogramowania.

Istotnym elementem dyscypliny wdrożenia jest przygotowanie materiałów dla użytkowników oraz przeprowadzenie szkoleń dla przyszłych użytkowników. Forma materiałów oraz szkoleń jest uzależniona od wybranej strategii wdrożenia oraz rodzaju zbudowanego systemu. Podstawowym celem szkolenia jest nauka zasad działania i operowania systemem. Może to być realizowane poprzez dostarczenie użytkownikom podręczników albo poprzez wbudowane w system materiały szko leniowe

(np. pomocniki kontekstowe). Szkolenia mogą być prowadzone na zasadzie bezpośredniego mentoringu w środowisku produkcyjnym lub być wykonywane na bazie przygotowanych prezentacji szkoleniowych oraz pracy w osobnym środowisku szkoleniowym. Efektem pomyślnego wdrożenia systemu jest działający system, sprawnie używany przez jego użytkowników. Wdrożenie powinno również dostarczyć ocenę zbudowanego i wdrożonego systemu (ewentualnie jego wersji pośredniej). Jest to podstawą do dalszych działań w ramach obecnego projektu (kolejne wersje przyrostowe), w następnych projektach (rozbudowa systemu w przyszłości) oraz w ramach dyscypliny konserwacji.

Dyscyplina konserwacji

Czynności konserwacji, zwane także czynnościami utrzymania, stanowią zawsze ostatni etap w cyklu życia oprogramowania. Obejmują one zapewnianie poprawnego funkcjonowania systemu w środowisku biznesowym. W trakcie pracy z systemem oprogramowania mogą zostać wykryte wcześniej nieznane defekty. Wymagana funkcjonalność lub cechy jakościowe systemu oprogramowania, które zostały zdefiniowane w ramach analizy wymagań, mogą ulec zmianie już po oddaniu systemu do użytku. Mogą się także zmienić samo środowisko i procesy biznesowe (np. zmiany organizacyjne w firmie, wdrożenie zmian w innym systemie, zmiana otoczenia prawnego). Wszystkie te sytuacje wymuszają podjęcie pewnych działań, które można podzielić na cztery grupy:

- czynności naprawcze obejmujące usuwanie defektów oprogramowania;
- czynności adaptacyjne dostosowujące oprogramowanie do zmieniającego się środowiska;
- czynności ulepszające wprowadzające nowe funkcjonalności i zmianę już istniejących;
- czynności prewencyjne przygotowujące oprogramowanie do przyszłych modyfikacji.

W niektórych przypadkach wprowadzenie zmian w funkcjonującym systemie może wymagać przejścia przez wszystkie dyscypliny, od wymagań do implementacji i wdrożenia. Wszelkie działania konserwacyjne powinny być starannie dokumentowane. Pozwala to unikać błędów we wprowadzaniu zmian i czyni system oprogramowania przejrzystym. Podjęcie wyżej wymienionych czynności powinna poprzedzić analiza kosztów i ryzyka wprowadzania zmian w funkcjonującym systemie. W wyniku takiej analizy może się okazać, że bardziej opłacalne od utrzymania bieżącego systemu będzie wytworzenie nowego systemu. W takiej sytuacji kończy się cykl życia oprogramowania, co prowadzi do rozpoczęcia kolejnego cyklu budowy i życia oprogramowania.